ЛАБОРАТОРНАЯ РАБОТА №3. РЕШАЮЩЕЕ ДЕРЕВО. КОМПОЗИЦИИ РЕШАЮЩИХ ДЕРЕВЬЕВ. МНОГОСЛОЙНЫЙ ПЕРЦЕПТРОН

(Продолжительность лабораторного занятия – 4 часа)

А. НАЗНАЧЕНИЕ И КРАТКАЯ ХАРАКТЕРИСТИКА РАБОТЫ

В процессе выполнения настоящей работы закрепляются знания студентов по разделам «Решающее дерево и композиции решающих деревьев», «Градиентный бустинг» и «Нейронные сети» курса «Применение методов искусственного интеллекта в электроэнергетике». Работа имеет экспериментальный характер и включает анализ данных и работы алгоритмов машинного обучения.

Целью работы является получение практических навыков работы с моделями решающего дерева, композиций решающих деревьев и нейронных сетей в программной среде Python.

Б. СОДЕРЖАНИЕ РАБОТЫ

Работа содержит:

1. Анализ, предварительную предобработку и визуализацию данных.
2. Обучение и применение моделей решающего дерева, случайного леса, градиентного бустинга и нейронных сетей с оптимальными параметрами на двух наборах данных.
3. Построение и визуализация метрики качества для обучающего набора данных и тестового.

Работа выполняется на компьютерах в интерактивной среде разработки JupyterLab.

В. ЗАДАНИЕ НА РАБОТУ В ЛАБОРАТОРИИ

1. Загрузить набор данных «income.csv».
2. Проанализировать загруженный набор данных:

2.1 определись признаки, в которых есть пропущенные значения, посчитать количество пропущенных значений по каждому признаку;

2.2 построить гистограмму объектов по признаку «workclass»;

2.3 визуализировать совмещенные гистограммы объектов по признаку «income» для двух значений признака «sex» на одном графике;

2.4 визуализировать совмещенные гистограммы объектов по признаку «income» для всех значений признака «race» на одном графике;

2.5 визуализировать совмещенные гистограммы объектов по признаку «workclass» для двух значений признака «income» на одном графике;

2.6 визуализировать совмещенные гистограммы объектов по признаку «workclass» для двух значений признака «sex» на одном графике;

2.7 визуализировать гистограмму объектов по признаку «age»;

2.8 визуализировать распределение объектов по признаку «age», используя «ящик с усами»;

2.9 визуализировать два «ящика с усами» по признаку «age» для двух значений признака «income» на одном графике;

2.10 визуализировать четыре «ящика с усами» по признаку «age» для двух значений признака «income» и двух значений признака «sex» на одном графике;

2.11 визуализировать «ящики с усами» для каждого из значений признака «race» по признаку «age»;

2.12 визуализировать тепловую карту корреляции признаков;

2.13 если в наборе данных пропущенные значения обозначены специальным символом, замените значения в таких ячейках на тип NaN;

2.14 определить категориальные признаки в наборе данных;

2.15 определить числовые признаки в наборе данных.

3. Подготовить и разделить исходный набор данных на тренировочный и тестовый наборы, в качестве целевой переменной возьмите признак «income».

4. Обучить модель решающего дерева для задачи классификации, построить графики зависимости F-меры на обучающей выборке и на тестовой от глубины дерева. Найти оптимальную глубину дерева, варьируя ее в выбранном диапазоне. Построить для оптимальной модели матрицу ошибок.

5. Обучить модель случайного леса для задачи классификации, построить графики зависимости F-меры на обучающей выборке и на тестовой от количества деревьев в композиции. Найдите оптимальное количество деревьев в композиции, варьируя их в выбранном диапазоне. Построить для оптимальной модели матрицу ошибок.

6. Обучить модель градиентного бустинга для задачи классификации, построить графики зависимости F-меры на обучающей выборке и на тестовой от количества деревьев в композиции. Найдите оптимальное количество деревьев в композиции, варьируя их в выбранном диапазоне. Построить для оптимальной модели матрицу ошибок.

7. Обучение модели многослойного перцепторна:

7.1 подготовить данные для обучения нейросети;

7.2 обучить модель многослойного перцептрона для задачи классификации с оптимальными параметрами, построить графики зависимости F-меры на обучающей выборке и на тестовой от количества эпох обучения. Определите оптимальное количество эпох обучения и

архитектуру нейросети. Построить для оптимальной модели матрицу ошибок.

8. Повторите пункты 6-7 для набора данных MNIST.

Г. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К РАБОТЕ В ЛАБОРАТОРИИ

К пункту 1.

Для того, чтобы загрузить данные в формате «.csv» используйте метод read_csv библиотеки Pandas, аргументом которого является путь к файлу. Метод возвращает объект класса DataFrame.

К пункту 2.

Пункт 2.1: воспользуйтесь методами isnull() и sum() класса DataFrame:

```
data = pd.read_csv('income.csv')
data.isnull().sum()
```

Пункт 2.2: воспользуйтесь методом класса hist() DataFrame (не забудьте подписывать графики и устанавливать корректные размеры визуализируемых графиков):

```
data['workclass'].hist()
```

Пункт 2.3: воспользуйтесь библиотеками Seaborn и Matplotlib для визуализации совмещенных графиков:

```
import matplotlib.pyplot as plt
import seaborn as sns
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.countplot(x="income", hue="sex", data=data, palette="Set1")
ax.set_title("Frequency distribution of income variable wrt sex")
plt.show()
```

Пункт 2.7: воспользуйтесь библиотеками Seaborn и Matplotlib для визуализации гистограммы:

```
f, ax = plt.subplots(figsize=(10,8))
x = data['age']
ax = sns.distplot(x, bins=10, color='blue')
ax.set_title("Distribution of age variable")
plt.show()
```

Пункт 2.8: воспользуйтесь библиотеками Seaborn и Matplotlib для визуализации «ящика с усами»:

```
f, ax = plt.subplots(figsize=(10,8))
```

```
x = data['age']
ax = sns.boxplot(x)
ax.set_title("Visualize outliers in age variable")
plt.show()
```

Пункт 2.9: воспользуйтесь библиотеками Seaborn и Matplotlib для визуализации «ящиков с усами»:

```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.boxplot(x="income", y="age", data=data)
ax.set_title("Visualize income wrt age variable")
plt.show()
```

Пункт 2.10: воспользуйтесь библиотеками Seaborn и Matplotlib для визуализации «ящиков с усами»:

```
f, ax = plt.subplots(figsize=(10, 8))
ax = sns.boxplot(x="income", y="age", hue="sex", data=data)
ax.set_title("Visualize income wrt age and sex variable")
ax.legend(loc='upper right')
plt.show()
```

Пункт 2.12: воспользуйтесь методами класса DataFrame для построения тепловой карты:

```
data.corr().style.format("{:.4}").background_gradient(cmap=plt.get_cmap('coolwarm'), axis=1)
```

Пункт 2.13: воспользуйтесь методом replace() класса DataFrame:

```
data.replace(' ?', np.NaN, inplace=True)
```

Пункт 2.14: воспользуйтесь генератором списка для определения категориальных признаков:

```
categorical = [var for var in data.columns if data[var].dtype=='O']
data[categorical].head()
```

К пункту 4.

Модель решающего дерева импортируйте из библиотеки Scikit-Learn:

```
from sklearn.tree import DecisionTreeClassifier
```

Глубину дерева выберете, установив значение атрибута конструктора класса max_depth.

К пункту 5.

Модель случайного леса импортируйте из библиотеки Scikit-Learn:

```
from sklearn.ensemble import RandomForestClassifier
```

Количество деревьев в композиции выберете, установив значение атрибута конструктора класса n_estimators.

К пункту 6.

Модель градиентного бустинга импортируйте из библиотеки CatBoost:

```
from catboost import CatBoostClassifier
```

Количество деревьев в композиции выберете, установив значение атрибута конструктора класса n_estimators.

К пункту 7.

Пункт 7.1: прежде, чем обучать модель нейросети, необходимо заполнить пропущенные значения в данных. Сделать это можно, выбрав наиболее часто встречающиеся значения признаков в качестве замены пропускам:

```
X['workclass'].fillna(X['workclass'].mode()[0], inplace=True)

X['occupation'].fillna(X['occupation'].mode()[0], inplace=True)

X['native_country'].fillna(X['native_country'].mode()[0], inplace=True)
```

Далее необходимо привести данные к типу NumPy - numpy.ndarray:

```
X_train = X_train.to_numpy(dtype=('float32'))

X_test = X_test.to_numpy(dtype=('float32'))
```

Для задачи бинарной классификации, необходимо выразить вектор бинарных ответов двумя бинарными векторами, каждый из которых будет содержать в ячейках 1 и 0 для отражения принадлежности объекта конкретному классу:

```
from tensorflow.python.keras.utils import np_utils

y_train = np_utils.to_categorical(y_train, 2)

y_test = np_utils.to_categorical(y_test, 2)
```

Помните о том, что для линейных моделей и нейросетей важно приводить признаки к одному масштабу.

Пункт 7.2: Импортируйте составные части нейросети из библиотеки TensorFlow (пример ниже приведен для версии 2.0):

```
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation, Dropout

from tensorflow.python.keras.utils import np_utils
```

Выберете последовательную модель сборки и соберите архитектуру нейросети:

```
NB_CLASSES = y_train.shape[1]
```

```
INPUT_SHAPE = (X_train.shape[1],)
model = Sequential()
model.add(Dense(32, input_shape=INPUT_SHAPE))
model.add(Activation('relu'))
model.add(Dropout(0.3))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(8))
model.add(Activation('relu'))
model.add(Dense(NB_CLASSES))
model.add(Activation('softmax'))
model.summary()
```

Скомпилируйте модель:

```
model.compile(loss='binary_crossentropy',
        optimizer = 'adam',
        metrics=['Precision', 'Recall'])
```

Запустите процесс обучения нейросети:

```
EPOCHS = 30
history = model.fit(X_train, y_train,
            batch_size = 32, epochs = EPOCHS,
            verbose = 1, validation_data = (X_test, y_test))
```

После обучения можно посчитать значение F-меры:

```
f1_score_list_train = []
f1_score_list_test = []
for i in range(EPOCHS):
    f1_score_list_train.append(2 * history.history['Precision'][i] * history.history['Recall'][i] / (history.history['Precision'][i] + history.history['Recall'][i]))
    f1_score_list_test.append(2 * history.history['val_Precision'][i] * history.history['val_Recall'][i] / (history.history['val_Precision'][i] + history.history['val_Recall'][i]))
```

Прогноз по классам для построения матрицы ошибок:

```
y_pred = model.predict_classes(X_test)
```

К пункту 8.

Загрузите данные MNIST с помощью TensorFlow:

```
from tensorflow.keras.datasets import mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Визуализируйте один из объектов:

```
# pick a sample to plot
sample = 1
image = X_train[sample]
# plot the sample
fig = plt.figure
plt.imshow(image, cmap='gray')
plt.show()
```

Так как каждый объект представляет собой двумерный тензор 28x28 пикселей, его необходимо привести к одномерному вектору перед обучением моделей градиентного бустинга и нейросети:

```
X_train = X_train.reshape(X_train.shape[0], 28*28)
X_test = X_test.reshape(X_test.shape[0], 28*28)
```

При расчете F-меры для модели градиентного бустинга передайте в объект f1_score атрибут average со значением «micro»:

```
from sklearn.metrics import f1_score
f1_score(y_test, model.predict(X_test), average='micro')
```

Для обучения нейросети преобразуйте векторы ответов:

```
y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)
```

При компиляции нейросети выберете категориальную кроссэнтропию:

```
model.compile(loss='categorical_crossentropy',
        optimizer = 'adam',
        metrics=['Precision', 'Recall'])
```

Для более наглядной визуализации матрицы ошибок для многоклассового случая можно наложить на нее тепловую карту:

```
import seaborn as sns
confusion_matrix_ = confusion_matrix(y_test, y_pred)
```

```
cm = pd.DataFrame(data = confusion_matrix_, columns = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'], index = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'])

ax = sns.heatmap(cm, annot=True, fmt="d")
```

Д. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ОФОРМЛЕНИЮ ИСПОЛНИТЕЛЬНОГО ОТЧЕТА

Исполнительный отчет должен включать в себя:

- титульный лист с названием лабораторной работы и фамилией студента;
- цель лабораторной работы;
- листинг кода;
- результаты работы каждого пункта задания в виде графиков Matplotlib с подписанными осями;
- выводы о проделанной работе.